Keywords – Python, RPG, catalogue, essentials, pokemon, fangame, Napier, PC, Flask

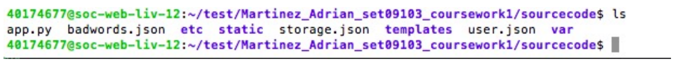
1 Introduction

My web-app title is **Pokemon Fan Website**. The main objective of this web application is to create a dynamic interface where users who love Pokemon can find Pokemon PC games designed by fans in a clear way. It provides knowledge to users regarding what Pokemon fan games have been released until now, what kinds of Pokemon[1] they would expect to find on those games and if either there is a demo version available to play or the game has already been finished. This compilation or gallery of files is something that hasn't been done previously and, therefore, would be visited as an encyclopedia for many followers of the franchise.

2 Design

2.1 Web Application Structure

The structural arrangement of the website files are as follows



```
40174677@soc-web-liv-12:~/test/Martinez_Adrian_set09103_coursework1/sourcecode$ ls
app.py  badwords.json  etc  static  storage.json  templates  user.json  var
40174677@soc-web-liv-12:~/test/Martinez_Adrian_set09103_coursework1/sourcecode$
```

Figure 1: **Development Server** - Files view on development server

- **app.py**:This contains all the server side code that controls all the front end layout
- An **etc folder** :This folder contains the configuration file **defaults.cfg** with static configuration parameters for the website
- 3 **Json files** which are mainly used for storing information and they include the **storage.json** where the pokemon fan information are stored,**user.json** which contains application user information and **badwords.json** which contains a list of prohibited words to be prohibited when posting a new pokemon on the website
- A **var folder**:This folder contains the **pokemonfan.log** files which contains application error logs
- A **templates folder**:This folder contains static website html documents for the front end of the website

2.2 General Web Application Design Overview

The web application design contains primarily all pokemon fan information with the ability to

- Search for pokemon
- Filtering of pokemon by status
- Registration of users
- Login into the website
- Adding a new pokemon.

[1]If unknown, check term in Appendix.

In terms of graphical design, this website is mainly structured by the Bootstrap front-end component library and a large number of CSS files that offer a unique style to each page. The first element designed was the navigation menu, which is a clear example of a simple design using the navigation bar with drop downs provided by default by Bootstrap. Some of the sites I took as a reference are the official forum of several PokÃl'mon fan sites[1] and a basic furniture catalog to obtain an idea of how to organise it.

The website "Coolors" that provides assistance showing the most complementary colors tonalities for a site was consulted [2]. The following colour palette is the result of the aforementioned site's generator:

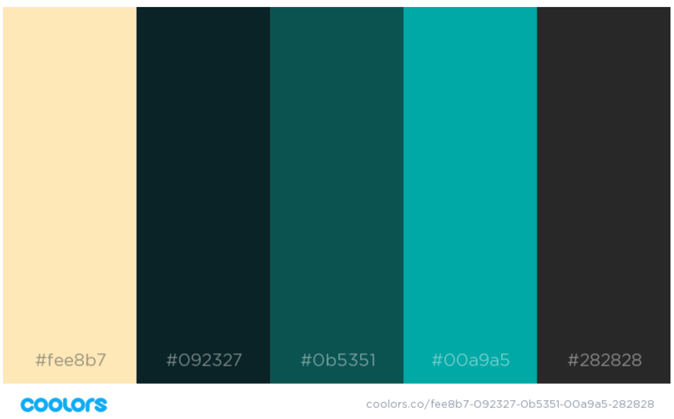


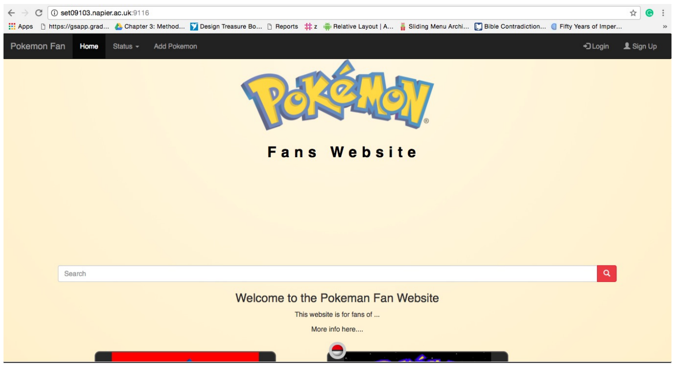
Figure 2: **Blue Hues Palette** - Yellow is used mostly for the background and black for the navigation bar and footer.

2.3 Home Page

The full gallery of all the games may be found on the home page ("/") with all its details displayed in advance without the need of clicking again and navigating to an individual page. The game characteristics that were deemed significant for the purpose of this project are their title, their status, the inclusion of Fakemon[1] with the usual creatures from the official games, the insertion of Mega Evolution and the first launch or publication of the main concept and plot. The home page contains the following:

- **Main navigation menu** which contains a category filter which divides the collection depending on its progress status. By clicking on it, you can choose between options: completed fan-games, games in demo stage and games that do not have any demo and are in developing phase. The main navigation menu also contains clickable buttons to navigate to the "home" page, "sign up" page, "login" page and the "Add Pokemon" page as seen below. This menu also appears on all other pages on the website
- A **search box** situated on the main page below some JavaScript animation, it searches for any subtext or full text that are matched by the word or sentence provided in the search box.

- A **footer** The home page also has a footer which can be seen on all the other pages. This footer shows itself when the user hovers the mouse over its area. The only element of the footer that is visible at first sight is the "pokeball" icon located above it.



Welcome to the Pokeman Fan Website

This website is for fans of ...

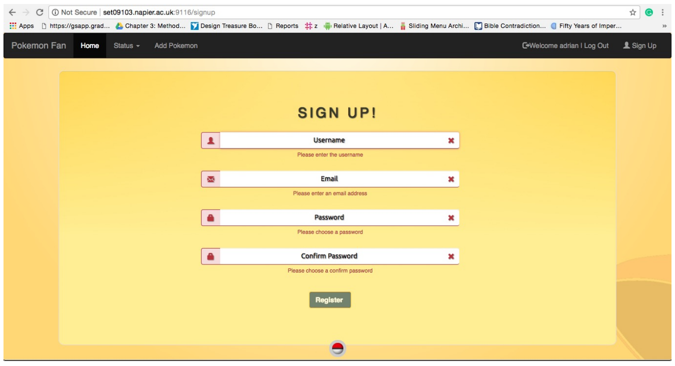
More info here....

Figure 3: **Home Page** - Website home design view

2.4 User Registration

The web application makes it possible for anyone to register. This is majorly needed for adding a new pokemon into the website. The user registration page prompts the user to provide a username, email, password and password confirmation. The following validations are implemented for this page:

Required Field: This validates that all the fields in the form has to be filled (see figure 4)



Username

Please enter the username

Email

Please enter an email address

Password

Please choose a password

Confirm Password

Please choose a confirm password

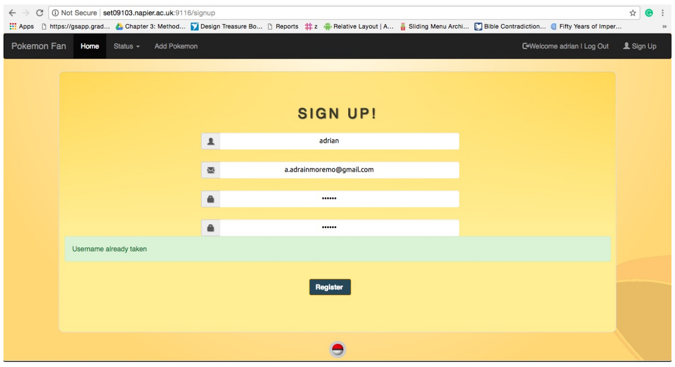
Figure 4: **Required Field Validation** - Form prompt stating that entering all the required fields is mandatory

Existing Username: This validates that the username is unique and hasn't been already taken (see figure 5)

Existing Email: This validates that the email is unique and hasn't been used before (see figure 6)

Password Mismatch: This validates that the password field and the confirm password fields are the same (see figure 7)

A successful log in redirects the user to the home page.

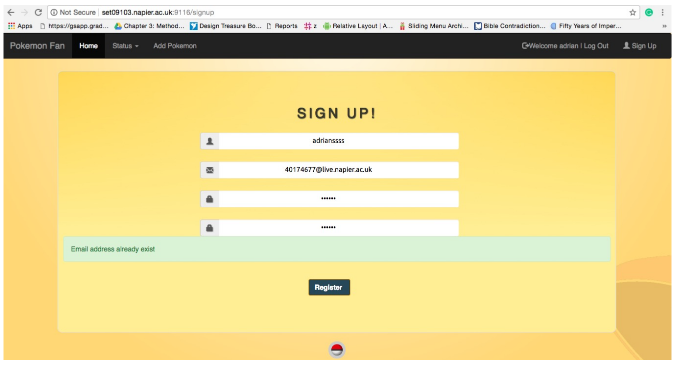


adrian

a.adrainmoremo@gmail.com

Username already taken

Figure 5: **Username Validation** - Form prompt stating that user name is already taken



adrianssss

40174677@live.napier.ac.uk

Email address already exist

Figure 6: **Email Validation** - Form prompt stating that email address already exists

2.5 User Login and SignOut

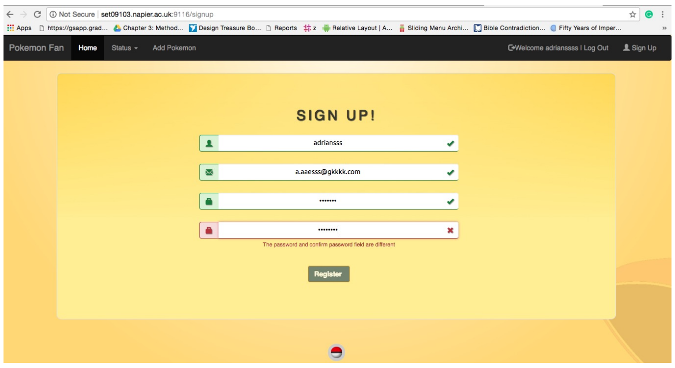
The login page mainly enables the already logged in user to log into the application. This is mainly needed when trying to add a new pokemon. The login page prompts the user accordingly if they have entered the wrong username and password as seen in figure 8 below. A successful login redirects the user to the home page of the web application

When a user is logged in, the application displays the username of the logged in user as well as a sign out button if they wish to sign out at the top right corner. A successful sign out redirects the user to the login page

2.6 Add Pokemon Page

The most important restriction is related to the "Add Pokemon" page. If the user has not logged in and selects the "Add Pokemon" button on the navigation menu, the user will be necessarily redirected to the login page. Once the user has signed in, he/she would be authorised to access the "Add Pokemon" page and add a brand new Pokemon fan-game. The Add pokemon page contains necessary validations that prevents addition of information in an unwanted way. The validations for these page are as follows:

- **Required Field**: This validates that all the fields in the form has to be filled (see figure 8)
- A **Badword Check**: This ensures that bad words are not included in the title, if such words are included, the form will not be submitted and the user is prompted

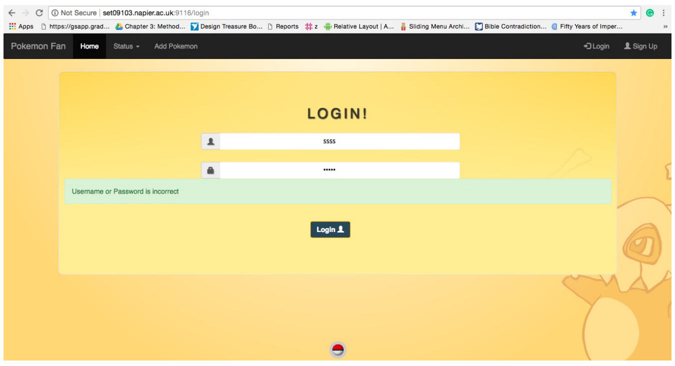

Not Secure | set09103.napier.ac.uk:9116/signup
Pokemon Fan
Home
Status
Add Pokemon
Welcome adrianssss I Log Out
Sign Up
SIGN UP!
Register

adriansss

a.aaesss@gkkkk.com

The password and confirm password field are different

Figure 7: **Password Mismatch Validation** - Form prompt stating that user should enter the same password for "password" and "confirm password" fields



ssss

Username or Password is incorrect

Figure 8: **Wrong Login Credential** - Form prompt the user that they have entered the wrong username or password

to remove such words from the tile (see figure 9)

- A **Duplicate Title Check**: This ensures that an already existing title is not duplicated on the website(see figure 10)

The user is prompted accordingly with a success message when a pokemon is added successfully.

2.7 404 page

Additionally, the web app also possesses 404 page that display a user friendly error message when a user enters a link that does not exist in the file system. See figure 11 below.

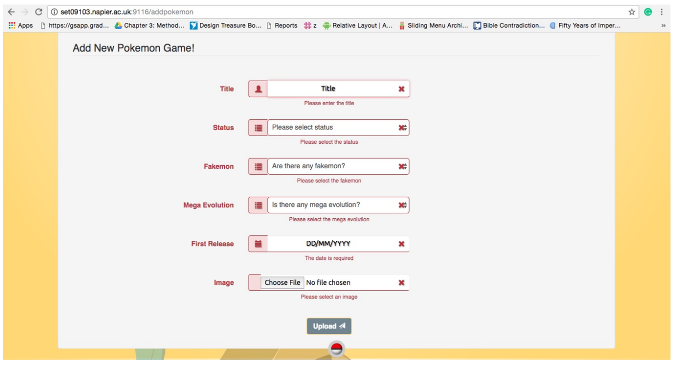
2.8 Other Project Artifacts

The project contains its configuration files in the **etc**. This is called defaults.cfg. This file contains valuable information about application parameters like the url, port number, ip address etc. The project also contains a log file in the **etc** folder called pokemonfan.log which logs application URL not found errors.

2.9 The Code Behind the Application

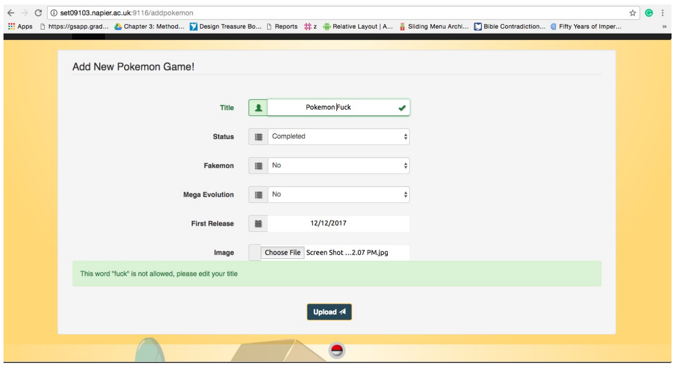
There are lots of code that drives the application, this section focuses on the most important ones. The main application loads the configuration file with the code snippet below

- **Logging Files Configuration**:



Title	Title
	Please enter the title
Status	Please select status
	Please select the status
Fakemon	Are there any fakemon?
	Please select the fakemon
Mega Evolution	Is there any mega evolution?
	Please select the mega evolution
First Release	DD/MM/YYYY
	The date is required
Image	Choose File No file chosen
	Please select an image

Figure 9: **Required Field Validation** - Form prompt to enter all required fields



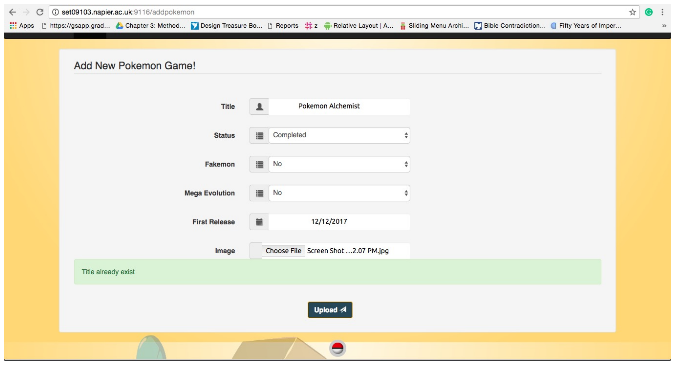
Title	Pokemon Fuck
Status	Completed
Fakemon	No
Mega Evolution	No
First Release	12/12/2017
Image	Choose File Screen Shot ...2.07 PM.jpg

This word "fuck" is not allowed, please edit your title

Figure 10: **Bad Word Validation** - Application prompting to remove bad word

Listing 1: Config and Log Code in app.py

```
def init(app):
     config = ConfigParser.ConfigParser()
     try:
          config_location = "etc/defaults.cfg"
          config.read(config_location)
          app.config['debug'] = config.get("config", "debug")
          app.config['ip_address'] = config.get("config", "ip_address")
          app.config['port'] = config.get("config", "port")
          app.config['url'] = config.get("config", "url")
          app.config['log_file'] = config.get("logging", "name")
     app.config['log_location'] = config.get("logging", "location")
     app.config['log_level'] = config.get("logging", "level")
     except:
          print "Could not read configs from: ", config_location
 def logs(app):
     log_pathname = app.config['log_location'] + app.config['log_file']
     file_handler = RotatingFileHandler(log_pathname, maxBytes=1024* 1024 * 10 , backupCount =1024)
     file_handler.setLevel( app.config['log_level'] )
     formatter = logging.Formatter("%(levelname)s | %(asctime)s |\
          %(module)s | %(funcName)s | %(message)s")
     file_handler.setFormatter(formatter)
     app.logger.setLevel( app.config['log_level'] )
     app.logger.addHandler(file_handler)eturn 0;
  }
```

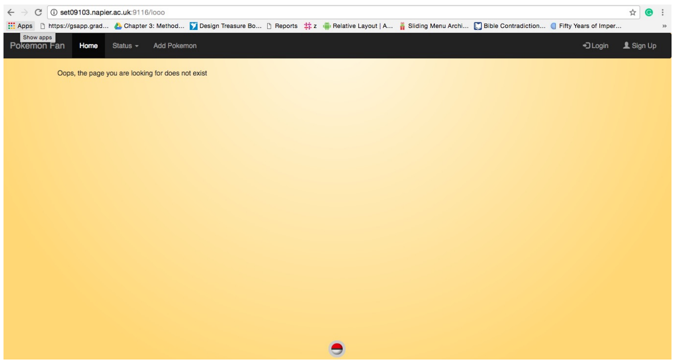

set09103.napier.ac.uk:9116/addpokemon

Add New Pokemon Game!

Title: Pokemon Alchemist
Status: Completed
Fakemon: No
Mega Evolution: No
First Release: 12/12/2017
Image: Choose File Screen Shot ...2.07 PM.jpg

Title already exist

Figure 11: **Existing Title Validation** - Website prompting that title already exist


set09103.napier.ac.uk:9116/looo

Oops, the page you are looking for does not exist

Figure 12: **Wrong URL** - Website prompts that the URL does not exist

The configuration file contains the following listing

Listing 2: defaults.cfg file

```
[config]
debug = True
ip_address = 0.0.0.0
port = 9116
url = http://set09103.napier.ac.uk:9116
[logging]
name=pokemonfan.log
location=var/
level=DEBUG

```

The application logs the error message for page not found errors using the code snippet below:

Listing 3: defaults.cfg file

```
@app.errorhandler(404)
def page_not_found(error):
    app.logger.info("From URL:"+request.path)
    return render_template('404.html')

```

and uses the configuration parameters in the application main method as seen below:

Listing 4: main

```

if __name__ == "__main__":
    init(app)
    logs(app)
    loaddata()
    app.run(host=app.config['ip_address'], port=int(app.config['port']), debug=app.config['debug'])

```

- **Home Page Files**: The home page uses a series of files which includes header.html (this contains the application header information as well as footer information that is inherited by every other page in the website). The home page list all available pokemon using the code snippet below

Listing 5: home

```
@app.route('/')
def home():
    return render_template('home.html', pokemonfans=pokemonfans)

```

The search button filters the pokemon by searching all the fields in the pokemon database that match the search criteria using the code snippet below

Listing 6: search

```
@app.route('/')
@app.route('/search', methods=['GET', 'POST'])
def search():
    if request.method == 'POST':
        pokemonsearch=[]
        what=request.form['search']
        what=what.lower()
        if what:
        #loop through the pokemon fans store and find any string that contains
        #what we are searching for and add it to pokemonsearch list
            for pokemon in pokemonfans:
                if what in pokemon["title"].lower():
                    pokemonsearch.append(pokemon)
                elif what in pokemon["status"].lower():
                    pokemonsearch.append(pokemon)
                elif what in pokemon["fakemon"].lower():
                    pokemonsearch.append(pokemon)
                elif what in pokemon["mega_evolution"].lower():
                    pokemonsearch.append(pokemon)
                elif what in pokemon["first_release"].lower():
                    pokemonsearch.append(pokemon)
            return render_template('home.html', pokemonfans=pokemonsearch)

        else:
            return redirect(url_for('home'))
    return redirect(url_for('home'))

```

The user status can also be filtered from the navigational menu. The code snippet below searches the status field from the database and filter according to whether they are in demo, developing or complemented as seen below:

Listing 7: filter status

```
@app.route('/filter')
def filterbystatus():
        pokemonfilter=[]
        status=request.args.get('status','')

        if status:
        #loop through the pokemonstore to find the pokemon with the specified status
            for pokemon in pokemonfans:
                if status in pokemon["status"].lower():
```

```
                                pokemonfilter.append(↩
pokemon)
                    return render_template('home.html',↩
pokemonfans=pokemonfilter)

                return redirect(url_for('home'))

```

- **User Registration, Login and Add Pokemon Forms**: These form all use the same technique, the form fields are validated by a bootstrap validation plugin which can be seen loaded in the header file below with some other css and javascript files

Listing 8: header.html snippet

```
<head>
 <link href="/static/bootstrap−3.3.7−dist/css/↩
bootstrap.min.css" rel="stylesheet" media="screen"↩
>
 <script src="http://code.jquery.com/jquery−latest↩
.js"></script>
 <script src="/static/bootstrap−3.3.7−dist/js/↩
bootstrap.min.js"></script>
 <link href="/static/css/custom.css" rel="↩
stylesheet" media="screen">
 <script src="/static/js/custom.js"></script>
 <link rel="stylesheet" href="http://cdnjs.↩
cloudflare.com/ajax/libs/jquery.bootstrapvalidator↩
/0.5.3/css/bootstrapValidator.min.css"/>
 <script type="text/javascript" src="http://cdnjs.↩
cloudflare.com/ajax/libs/jquery.bootstrapvalidator↩
/0.5.3/js/bootstrapValidator.min.js"></script>


```

This forms validates user data using the custom.js file located in the **static/js/** folder of the web application.For example, the sample validation below ensures that the mega evolution field when filling the add pokemon is not empty

Listing 9: custom.js snippet

```
mega_evolution: {
         validators: {
             notEmpty: {
                 message: 'Please select the mega ↩
evolution'
             }
         }
     }

```

The custom.js files also submits the form data to the file using ajax, a sample submission can be seen below:

Listing 10: custom.js logging ajax snippet

```
 $.ajax({
                       type : 'POST',
                       url : $form.attr('action'),
                       data: form_data,
                       datatype: 'json',
                       contentType: false,
                       processData: false,
                       success: function(data) {
                         if (data=="incorrect"){
                           $('#success_message').html("↩
Username or Password is incorrect");
                           $('#success_message').↩
slideDown({ opacity: "show" }, "slow"); // Do ↩
something ...
                         }else{
                           window.location.href="/";
                         }

                       }
                   });

```

3 Enhancements

I would like to have created a directory full of game files inside the repository so that the user would be able to download the games from the website. Thus, the subject wouldn't have the need of copying the game title and searching in any browser in order to download it. Nevertheless, due to size limitations, this option had to be discarded from the very beginning.

In addition, I would have liked to add further features and classifications to the games, such as external links to other pages and relevance forums.

Before starting the project, I was considering making a collection of something better known. For example, instead of having a catalog of Pokemon games, my intention was to create a group of the actual monsters of the game. Maybe I could have developed a web application involving both elements, Pokemon and RPG games, but such a modification would have drastically affected the size of the repository and the speed with which this website is loaded in the browser.

4 Critical Evaluation

4.1 Poor Features

Scarcity of visual animations for a younger audience. I can affirm with certainty that the currently presented version is not the best designed, given that there have been times when the forms had a unique appearance (especially due to the combination of several shades and opacity) and certain animations with a theme based on Pokemon. But due to certain setbacks with the functionality of those forms, the creative part was discarded and there was no other alternative but to start from scratch with those affected pages.

Dependence on external websites. If the development period had been longer, I would have searched for all the online libraries whose links are present in the source code inside the ¡head¿ tag in some HTML files and install them locally. The purpose of this action would be to refrain from connecting to Internet and avoid using that global system as an essential source of libraries.

Separation of permissions in the accounts. Another expectation I had was being able to establish numerous levels to the Log In system as currently happens with CMS[2] like Drupal. Therefore, taking into account that hypothesis, the owner of this web page would be the administrator, which would have permission to add any recent game, and ordinary users would only have permissions to read, but not to edit any type of content.

Lack of Delete and Edit options. No functionality to delete and edit elements has been inserted into the interface, the only way to perform that action would be manually through the JSON file, which is a clear disadvantage.

[2]If unknown, check term in Appendix.

Taking into account the defects of the website, certain tools will be extremely useful for those who visit the website.

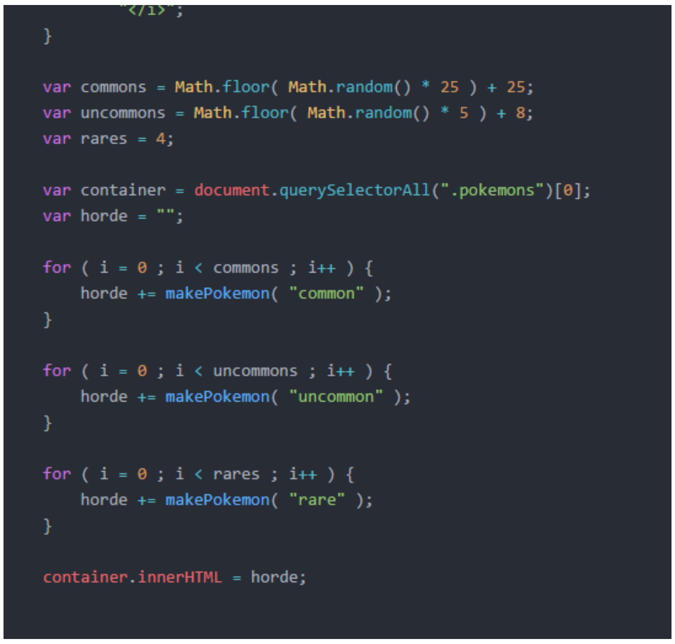
4.2 Strong Features

4.2.1 Search system

A customised system that allows the user to search for the fan-game they want to play.

4.2.2 JavaScript animation

This animation is based on the anime of the Pokemon franchise and works thanks to a harmonious combination of movement transactions in CSS and various JavaScript functions that allow a coordinated timing.



```
        "</i>";
}

var commons = Math.floor( Math.random() * 25 ) + 25;
var uncommons = Math.floor( Math.random() * 5 ) + 8;
var rares = 4;

var container = document.querySelectorAll(".pokemons")[0];
var horde = "";

for ( i = 0 ; i < commons ; i++ ) {
    horde += makePokemon( "common" );
}

for ( i = 0 ; i < uncommons ; i++ ) {
    horde += makePokemon( "uncommon" );
}

for ( i = 0 ; i < rares ; i++ ) {
    horde += makePokemon( "rare" );
}

container.innerHTML = horde;
```

Figure 13: **JS code to generate a random pokemon horde** - Every time the main page is refreshed, a distinct group of creatures will appear.

4.2.3 Difficulty

In terms of difficulty, the simplicity and clarity of my website stands out. Perhaps the positioning of the filter within the navigation menu is not the most conventional way of organising a catalog, but on the other hand, the empty space of the main menu should have been filled with any other type of content anyway.

5 Personal Evaluation

5.1 Learning Outcomes

My greatest achievement has undoubtedly been the expertise acquired in the use of client side's scripting languages. The use of HTML, CSS and JavaScript as a group has required close observation of those elements linked between the files of each programming language. For example, many transitions and effects of the main page's animation in JavaScript took as identification the classes or IDs declared in the "home.html" file. In turn, these divisions had been graphically altered by a couple of CSS files.

5.2 Challenges

The biggest challenge I've come across has been keeping the code simple, that is, not including anything additional to what was learned in class. This issue could be defined as a "restriction of available tools", but should not be considered for the next assignment since the second coursework offers a greater flexibility. The initiative to carry out supplementary research, which is as beneficial as harmful, can lead to long and time-consuming Internet searches. As an evidence of what has been previously mentioned, this is the code of an older version of the "app.py" file. It's possible to observe that the application was slightly dissimilar and that there was no JSON files involved. The data of games and users were stored in a Python file instead[3].

```
    [caption = "app.py" with classes to validate forms included]
import logging,pickle,ConfigParser
from logging.handlers import RotatingFileHandler
from flask import Flask, render_template, url_for, request,redirect↩
    ,flash,session
from wtforms import Form, BooleanField, StringField, ↩
    PasswordField, validators
app = Flask(__name__)
app.secret_key='\xd9\xa8\xf5\xafm\xec\xa2J\x11`\x8fH\xbeO\↩
    xeb\x86\x05\xaf"\xfc\x1c}s\xe0'
pokemonstore=[]
pokemonusers=[]

```

5.3 Methods and Problem Solving

I made use of diverse information sources to solve problems and for other purposes. The main source of consultation for both design and Python-Flask has been **Stack Overflow**. It is a web platform especially useful to understand and study in depth the features of CSS alignment, usability guidelines and the clearest way to write the labels inside Login and Account Registration buttons.

5.4 Reflection on Performance

All my effort has been put into this project. Even the smallest detail has been taken into account to deliver a practical app. I collected a copious amounts of recommendations such as what terms is best to use for the form buttons, "sign in" / "sign out" or "log in" / "log out"? Taking into consideration the grading system, I believe there has been a slight imbalance in my criteria, focusing too much on the design while more features could have been added.

6 Conclusion

Making decisions for an individual project is complicated. A dynamic animation and beautiful design were "sacrificed" for a more accessible user interface and to corroborate that the general app performance is appropriate. I have used JS and CSS libraries with great efficiency and, although I have not been able to integrate them with my own code, the final result allows for a copious amount of user interactivity.

7 Appendix: Jargon and Unusual Terms

Pokemon: fictional creatures that humans collect, trade and train to battle each other[4].

Fakemon: fake Pokemon made and designed by fans.

CMS: Content Management System.

Mega Evolution: a special metgod to make your Pokemon stronger within the game.

References

[1] T. PokÃĺCommunityâĎć, “PokÃĺcommunity,” 2016.

[2] F. Bianchi, “Coolors,” 2016.

[3] A. Martinez, “Martinez$_{a}drian_{s}et09103_{c}oursework1,''October2017$.

[4] L. two, “PokÃĺmon,” October 2017.